FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Statistical Modelling of Microorganisms

Optionaler Untertitel der Arbeit

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Theresa Wihann
Matrikelnummer 1107772

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Dipl.-Ing. Dr.techn. Assoc. Prof. Ivan Viola
Mitwirkung: Dr.techn. Peter Mindek
Dipl.-Ing. Tobias Klein

Wien, 28. Februar 2017

Theresa Wihann

Ivan Viola

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Statistical Modeling of Microorganisms

Optional Subtitle of the Thesis

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Theresa Wihann
Registration Number 1107772

to the Faculty of Informatics

at the TU Wien

Advisor: Dipl.-Ing. Dr.techn. Assoc. Prof. Ivan Viola
Assistance: Dr.techn. Peter Mindek
Dipl.-Ing. Tobias Klein

Vienna, 28th February, 2017

Theresa Wihann | Ivan Viola

Erklärung zur Verfassung der Arbeit

Theresa Wihann
Rolandsberggasse 73/1 3400 Klosterneuburg

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 28. Februar 2017

Theresa Wihann

Acknowledgements

This bachelor thesis was supported by Ivan Viola, Tobias Klein and Peter Mindek. I would like to express my thanks to Ivan Viola for the continues impulses and his constructive critic. My thanks also goes to Tobias and Peter for the day and night support in mathematical and implementation related questions. Finally I want to express my thanks and appreciations to my colleague Lukas Mitterhofer, who perfectly complemented me.

Kurzfassung

Modelle von natürlichen Organismen können Wissenschaftler in diversen Bereichen wie der Medizin, Pharmazie oder Biologie unterstützen. Die meisten biologischen Organismen bestehen aus mehreren tausend Komponenten wie Proteinen, Lipiden und Säuren. Das Erstellen von solchen großen Strukturen ist oft sehr kompliziert und zeitaufwändig. In unserer Arbeit wollen wir daher einen auf statistischer Modellierung basierenden Ansatz vorstellen. Dieser ermöglicht die Modellierung von Mikroorganismen mit einem relativ geringen Aufwand. Die Modellierung basiert auf einem Entscheidungsbaum und verschieden Wahrscheinlichkeitsberechnungen. Es wird von einer zufälligen Verteilung der Objekte in der Szene ausgegangen. Danach werden sukzessive Informationen über die Veränderungen gesammelt, welche der User an den Objekte vornimmt. Diese dienen als Vorlage für die Veränderung der Orientierung und Position aller anderen Objekte in der Szene. Der Ansatz wurde praktisch in einem Tool umgesetzt, welches mit Unity3D entwickelt wurde.

Abstract

Models of a biological organism can support scientists in various fields, such as medicine, pharmacy or biology. When precise macromolecular composition of an organism is to be captured by the model, tens of thousands of protein macromolecules have to be positioned in order to create the model. This process is complicated and time consuming. In this thesis we propose an approach based on statistical modeling of microorganisms, which enables the creation of such scenes with minimal effort. The modeling is based on decision trees and probability calculations. The basic principle is to start with a random distribution. Then subsequently adjustments from the user on one or several objects, such as molecules, are gathered. These serve as examples to change the orientation and location of all other objects in the scene. The approach is realized in a tool implemented in Unity3D.

Contents

CHAPTER 1

Introduction

Visualization and creation of structures, which are invisible to the naked eye, is a challenge in various branches of natural science. Such visualizations can help scientists to disseminate their discoveries to the public, or communicate them to scientists from different fields. Computer graphic is often utilized to provide the presentation of these structures. In our approach we concentrate on the modeling of macromolecular scenes depicting microorganisms like viruses or bacteria. On the macromolecular scale, sometimes also referred to as mesoscale, a microorganism consists of proteins, lipids and nucleic acids, which are built of individual atoms. In order to model an organism on this scale, all the protein and other molecules have to be placed in space to create realistic representation of the given organism.

When creating a biological model one of the most important factors is, that the model is not too regular, since nature is not perfect. Most of the state of the art modeling concepts are rule based, which produces a kind of inflexibility that we want to address with the statistical method of modeling described in this thesis. As starting point we used a model of an HIV which is created with cellPACK [GTJO14] a software package for modeling microorganisms with an atomic resolution. Creating a complete model with cellPACK is time consuming and need additionally tools. We want to provide the whole modeling process in one tool in a reasonable amount of time, without the inflexibility of rule based approaches.

With our modeling approach we offer the possibility to create structures consisting of several thousand proteins interactively. We limit ourselves to the modeling with proteins only, since we use the protein models provided by the Protein Data Base[2]. For every protein the position of the center and local rotation is stored. The information of the structure of the protein comes from pdb files, which can be downloaded from the PDB for free. In order to give the user the opportunity to use every available protein of the

[2] http://www.rcsb.org

database, it is possible to download pdb files from the website while modeling by using the correct abbreviation of the protein in the protein database.

To model large biological macromolecular scenes, it is not useful or rather not possible to place every protein by hand. Therefore we propose a statistical approach, which is based on a decision tree where every node corresponds to an area of the scene and holds a probability value for the placement, rotation and cluster formation of the proteins in this area. The decision tree is created during the modeling process and any action of the user affects the structure or the probabilities. After every change made by the user, the positions and rotations of the affected proteins are recalculated based on all previous actions. This enables the user to create extensive scenes. Our approach deals with the calculation of the placement and orientation of the proteins and not with the rendering. To achieve a fast rendering of large biological data we integrated the practical implementation in cellView [MAPV15].

CHAPTER 2

Related Work

In this section we describe the work related to statistical modeling, methods or tools to create very large structures or scenes and mathematical concepts used in our approach.

State of the art in the field of modeling of microorganisms is cellPACK [GTJO14], which is a software package for modeling microorganisms with an atomic resolution. It populates predefined geometric compartments with models of protein molecules, based on a so-called recipe file. The recipe contains information, such as the hierarchy of the model and protein related information, such as the amount of each protein type or pdb abbreviation. The models of the proteins are obtained from the Protein Database (PDB). To create a final model of a microorganism additional tools are required. The creation of a model of a microorganism with cellPACK is cumbersome and quite time consuming, since it is necessary to manually create the recipe file, and the model generation itself is computationally expensive and it can take up to an hour. Therefore, we propose a statistical modeling system, which provides the possibility to model organisms visually, in a WYSIWYG (what you see is what you get) manner, while the computation is carried out in real time, thus significantly lowering the model creation time.

Statistical modeling and generation approaches are used in different fields. Conklin at al. [Con03] uses an approach to generate music from statistical models. The statistical models to classify new pieces or sequences of pieces, are created in advance based on existing pieces. The generation of text [Lan00] is another field where statistical methods are constantly used. With the statistical modeling of microorganisms we apply the statistical approaches in a different context and field. Our approach includes several components. The central component is the decision tree for the placement and orientation of the proteins.

Decision trees are used in various areas, such as data-mining [AW97], machine learning [SL91], agent-based modelling [CCW98], and decision theory [BDW02]. Grabczewski at al. [Grą14] give an overview of methods to generate decision trees like CHAIDs ,CARTs,

ID3 and C4,5. In our approach the created decision trees are on the one hand rather flat, since every compartment corresponds to one node and the amount of compartments corresponds to the biological structure of a cell and on the other hand mostly no binary trees, since a compartment can have an arbitrary amount of sub-compartments.

Mathematically the rotation can be implemented in different ways [Die06]. The main mathematical approaches to represent rotations are rotation matrices, Euler angles and quaternions. The most common method is the set of three Euler angles which describes the rotation around x, y and z axis. Euler angles are really intuitive in the case that the rotation should be around the x, y or z axis. When the rotation should be performed around arbitrary axis it is quite complicated. Furthermore the order of the rotations is not arbitrary a rotation in sequence x,y,z provides a different output than y,z,x. Another problem which occurs in connection with Euler angles is the gimbal lock [Han05]. Quaternions are a set of four values and basically consists of a rotation axis and a rotation angle. The representation as quaternion is not as intuitive as Euler angles but it avoids the gimbal lock, it is a rotation around one single axis and for complex rotation faster. We wanted the advantage of intuitive calculation with quaternion as well as avoiding the gimbal lock, but we wanted the intuitive setup for rotations. Therefore we decided for the internal representation as quaternions, but for the user interface we took the Euler angles. It is necessary to control the 3D rotation, there exist several approaches for that purpose [BRP05] like Chen et al.'s Virtual Trackball,Two-Axis Valuator with Fixed Up-vector and Bell's Virtual Trackball. In our approach we used the built-in handles from unity to control the rotation.

In our approach, the user provides several samples of how the protein molecules should be oriented. The orientations of all other proteins has to be interpolated from these samples for a naturally looking model. Silverman at al. [Sil86] discuss various methods for the density estimation, such as histograms, orthogonal series estimators and the kernel density estimation. In our approach we used the most commonly used estimator, the kernel density estimation [Par62] to interpolate between the samples. The KDE provides a continuous function, which can be calculated in every point, it is easy to implement and it is quite simple to create a multivariate KDE. Which is necessary since the sample quaternions consist of four values.

Our tool is integrated into cellVIEW [MAPV15]. CellVIEW is a tool for real-time visualization of biological models with atomic resolution, employing various acceleration methods and level-of-detail schemes. The models visualized by cellVIEW are static representations of the microorganisms in a particular point in their lifetime. It is possible to examine the organism by zooming and applying cutaway views, as well as using an advanced visibility control called *Visiblity Equalizer* [LMMS+16]. The user also gains information about the proteins which the model consists of. However, it is not possible to interactively create a model or modify an existing model in the tool. In this thesis, we propose a system which provides such functionalities.

CHAPTER 3

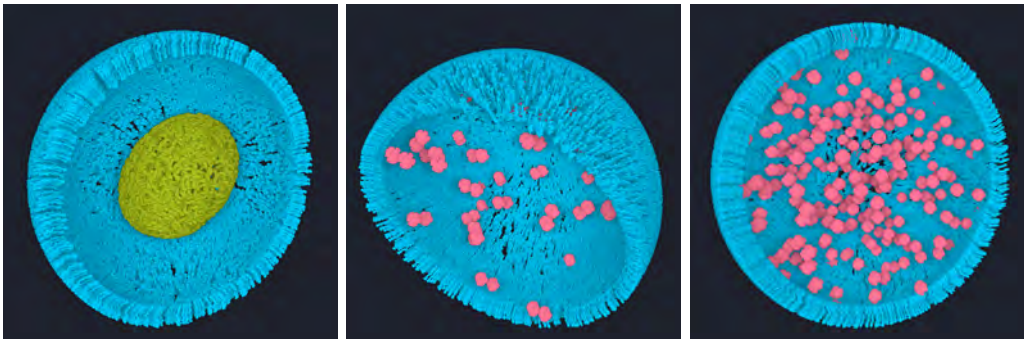
Concept and Methods

Every scene created with our tool consists of different proteins. The proteins build compartments (see Figure 3.1a), clusters (see Figure 3.1b) or can just be distributed freely in space (see Figure 3.1c). Compartments are areas of the scene enclosed by a membrane consisting of lipids, in our approach represented by protein. For instance, in a cell, the compartments would correspond with the organelles, such as nucleus or mitochondria. Protein behavior often requires relative positioning to the membrane, such as being close to a membrane or following a gradient. For that reason we use in our approach *distance functions* to describe compartments, whose 0 level set correspond with the compartment surface. The input for the function is a 3D point and the result is a positive or negative decimal. A negative decimal implies, that the point is inside the compartment and a positive that it is outside. The value of the decimal indicates the distance to the surface. The surface is populated by proteins of one type, chosen by the user.

The user has the opportunity to form clusters with an arbitrary amount of different or similar proteins. The center of the cluster is the mean of the protein centers. The clustered proteins are placed, with the same distance, randomly around the center. Each time the user adds another protein to a cluster, all clusters are recalculated and distributed again.

For the placement of the proteins or clusters there exist three regions: It can be inside, outside or on the surface of a compartment. One compartment object can include an arbitrary amount of child compartments.

The modeling process is divided into two phases, the compartment creation and the scene population with proteins. During the first phase of the process, the compartment creation, it is possible to choose one of the provided distance functions and the type and amount of proteins to populate the surface. In our approach the distance function for ellipsoids and for capsules are implemented, since these forms are appropriate to



(a) compartments (b) pair cluster (c) freely distributed

Figure 3.1

approximate natural shapes. It is possible to scale, rotate and move compartments. The only limitation is, that the outermost compartment can not be moved outside of the center of the scene. In this phase the rotation of the surface proteins can be changed as well. When the first phase is finished, the structure of the decision tree will be created. In the decision tree every compartment object of the scene corresponds to one node of the tree and every node contains the possibilities for the placement, rotation and clustering for the respective compartment. After the first phase it is not possible to change the shape and position of the compartments or the rotation of the surface proteins again.

In the second phase the scene can be filled with proteins. The amount and the type of the added proteins can be set by the user. Once they are added the user can start to change the rotation, the position and the cluster formation of all proteins of the same type by changing one of them. Furthermore, in the second phase the probabilities of the decision tree are influenced. In both phases it is possible to download pdb files with structural information of the proteins needed for the microorganism. In order to give the user more control over the outcome, the color and description of the proteins can be changed. This information is stored in a json file.
It is possible to export new scenes and import scenes, which have been modeled before. The exported file contains the information of the decision tree, compartments and proteins to recreate the scene. The user can continue working with the imported decision tree.

3.1 Decision tree

In order to generalize the user affected modification across the compartments hierarchy, we needed a structure that capture the changes of the placement, rotation and cluster formation. In our approach we use as structure a decision tree, since we have different protein types we need one per type. The calculation of position and orientation of the proteins in the scene as well as the cluster membership is based on these decision trees. The decision tress are generated and transformed during the modeling process. The structure of different trees is always the same, where every tree-node corresponds to

one compartment and contains the probabilities for the placement and rotation of the proteins inside of this compartment. The tree for a protein type is created as soon as the user performs the first modification for that protein type.

Every action of the user affects the structure of the tree or the probabilities for rotation, placement and clustering. The probability for the placement inside a compartment is simply the amount of user placed proteins inside of the compartment divided by the amount of all user placed proteins in the scene. The cluster formation process as well as the calculation of the probability and determination of rotations will be discussed in the next sections.

3.1.1 Clustering

In nature not all proteins are distributed freely in a cell, some are forming clusters. So we give the possibility to form cluster, either with proteins of the same type or with proteins of different types. To provide the functionality we had to distinguish between two different kinds of movement. First the movement across compartment borders, which affects the placement probabilities and second the movement within a compartment which leads to cluster formation. These assumptions were taken, because the movement inside a compartment would not have any other useful purpose, since the positions are recalculated after each user affected movement. To build a cluster the user has to move one protein towards another protein, with which one the cluster should be formed. After the user influenced action all affected proteins in the compartment form cluster. For example if the user moves a protein towards another protein, of the same type, to form a pair, all proteins of the same type, in the current compartment, will follow and form pairs. If the compartment contains a odd amount of proteins the remaining protein will be placed as non clustered protein. The amount of proteins in a cluster is not restricted. The clusters are distributed uniformly in the current compartment, it is not possible to change the position of a whole clusters. To dissolve a cluster or to reduce the amount of proteins in a cluster the user has to move one of the clustered proteins a certain distance away from the cluster. The distance is twice the distance between the protein centers. The clustering information is stored separately,for every compartment, in the corresponding tree node.

3.1.2 Position and Rotation

Every single protein in the scene has its own rotation and position. The position is described by a 4D vector $[x, y, z, w]$ where x, y and z correspond to the position in the 3D space and w stores the unique ID of the protein type. The rotation is described as a quaternion, which represents a rotation as a set of 4 numbers. The quaternion, which is saved for every protein, describes the rotation in local space of the protein. When the proteins are added to the scene the rotation is $[0, 0, 0, 0]$. After the first user affected rotation proteins are always rotated relative to the surface of the closest compartment

object. To achieve that, it is necessary to use the normal vectors of the compartments as references.

The **normal vector** is calculated at the surface point closest to the currently focused protein. The normal vector is equal to the gradient, which indicates the direction of the biggest rate of increase of the function in the surface point. Since all compartments are described by distance functions, the gradient is calculated with the difference quotient with central difference (see equation 3.1). The central difference is the mean of forward- and backward difference.

$$\frac{\Delta y}{\Delta x} := \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \tag{3.1}$$

$\Delta x := x_1 - x$
x ... x value of closest point on the surface
x_1 ... x value of an adjacent point

Since the calculation is in 3D space it has to be calculated in x, y and z-direction. The difference quotient for x, y and z are the x, y and z components of the normal vector.

Quaternion

There are different methods [Die06] to work with rotations. As already mentioned in this work, in our approach rotations are described by quaternions. Quaternions are not as intuitive as Euler angles but much faster and easier to work with in complex operations. A problem which occurs in connection with Euler angles is so called gimbal lock. Since the rotation is specified by three angles around three perpendicular axes which are applied consecutively, it might happen that two of the axes get aligned after applying the rotation, causing a loss of one degree of freedom. Quaternions are independent of the coordinate system and the gimbal lock does not occur because the point, which should be rotated, is moved on the surface of a unit sphere located in the origin of the coordinate system. The quaternion represents a rotation as a set of 4 numbers $[x, y, z, w]$ in a range of $[-1.0, 1.0]$ and consists basically of a rotation axis and an angle.

Quaternion components:

$$x = rotationAxis.x * sin(\frac{rotationAngle}{2})$$
$$y = rotationAxis.y * sin(\frac{rotationAngle}{2})$$
$$z = rotationAxis.z * sin(\frac{rotationAngle}{2})$$
$$w = cos(\frac{rotationAngle}{2})$$

A quaternion is still readable, if it is defined by $[0.1, 0.7, 0.0, 0.7]$ it is possible to gather from this, that it is a rotation mostly around the y-axis. Because the x and z values are smaller than y. The transformed equation of w, $2 * arccos(w)$ gives the rotation angle in

radians. In case of 0.7 it is 1.57 which equals a rotation around 90° round the rotation axis. To cumulate two rotations, the quaternions are multiplied by each other. The multiplication is not commutative.

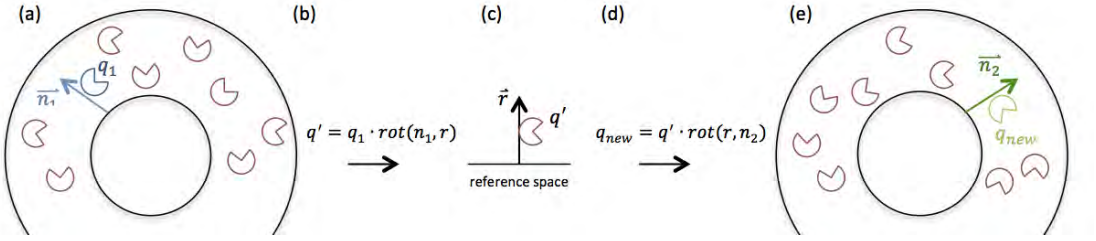


Figure 3.2: (a) initial situation, the blue protein is the user rotated one, with the normal vector n_1 and quaternion q_1; (b) transformation of q_1 in reference space; (c) transformed quaternion $q^{'}$ is stored; (d) transformation of $q^{'}$ to individual quaternion q_2, this calculation is performed for each affected protein; (e) final output with correct rotation of the proteins

To infer from the user input rotation to the rotation of every single protein in the scene the quaternions of the user-rotated proteins (reference protein) q_1 are stored as samples in the respective tree node, but not in the original form. As the rotation is relative to a compartment, the rotation quaternion of every protein is different. Because of this the quaternion q_1 of the reference protein has to be transformed to a reference space shown in Figure 3.2. To achieve that, we calculate the rotation between the normal vector at the surface point closest to the reference protein and the vector $r(0, 1, 0)$ which corresponds to the standard y-Axis. The original quaternion q_1 of the protein is multiplied with that rotation which transforms it into reference space. The sample quaternions in reference space are the base of the calculation of the rotations for every single protein. For the calculation of the individual rotation for all proteins in the scene the corresponding normal vector and the rotation quaternion between the normal and the reference vector r is calculated and multiplied with the reference quaternion $q^{'}$.

3.2 Kernel Density Estimation

If the proteins have the same rotation, relative to a compartment surface, the output is often to structured for a model of a biological organism. To achieve a model that is not too structured the reference quaternions are created randomly and are evaluated with a *kernel density estimation* (KDE) with a Gaussian kernel, based on the sample quaternions. The KDE is a statical method to estimate the probability distribution of a random value. For the KDE different kernels can be used, in our tool a Gaussian function (Equation (3.2)) is implemented. The bandwith h has an impact on the smoothness of the KDE. The higher it is the lazier the KDE reacts to individual samples. In our approach the bandwith is set to a fixed value and not changeable by the user. The

variance for the KDE can be set by the user for the particular protein types. For better usability the variance is specified in degrees by the user and converted to radians before the calculations. The variance is the expected squared deviation from the stored samples. The variance in x, y and z direction is the same. The user can change the variance of the currently selected protein at any time, but it only takes an effect, if the protein is also rotated in the same process step.

$$g(x) = a \exp(-\frac{(x-b)^2}{2c^2}) \tag{3.2}$$

Since the quaternion consists of four values it is necessary to use a 4D Gaussian function as kernel for the kernel density estimation.

$$g(x,y,z,w) = a \exp(-(\frac{(x-b_{\mathrm{x}})^2}{2{c_{\mathrm{x}}}^2} + \frac{(x-b_{\mathrm{y}})^2}{2{c_{\mathrm{y}}}^2} + \frac{(x-b_{\mathrm{z}})^2}{2{c_{\mathrm{z}}}^2} + \frac{(x-b_{\mathrm{w}})^2}{2{c_{\mathrm{w}}}^2})) \tag{3.3}$$

With a= $\frac{1}{\sigma\sqrt{2\pi}}$, $b = \mu$ and $c = \sigma$ it results in the correct KDE for quaternions:

$$f(x,y,z,w) = \frac{1}{nh} \sum_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2\sigma^2}((x-x_{\mathrm{i}})^2 + (y-y_{\mathrm{i}})^2 + (z-z_{\mathrm{i}})^2 + (w-w_{\mathrm{i}})^2)) \tag{3.4}$$

The result of the equation is the probability for the quaternion $[x, y, z, w]$ to be accepted as rotation for a protein. The algorithm to create a random Gaussian distributed quaternion with kernel density estimation is implemented as follows:

Algorithm 3.1: create random quaternion

Input: a List with quaternions samples, a skalar variance
Output: quaternion q

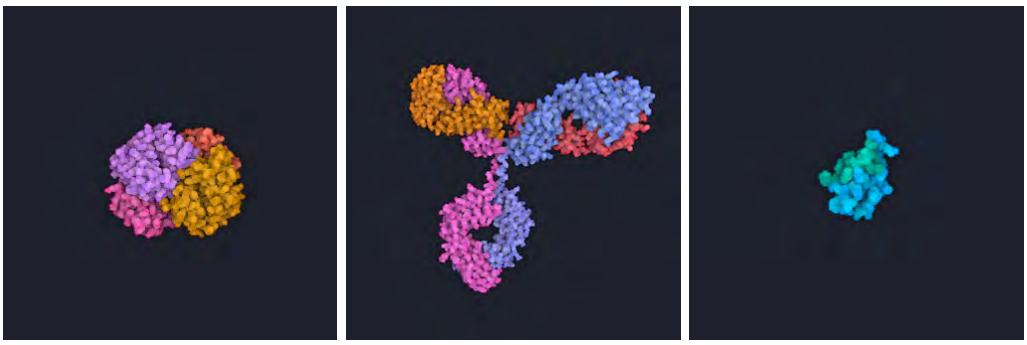
$\sigma = \frac{2\pi}{360} * variance$;
factor = $\frac{1}{n0.2\sigma\sqrt{2\pi}}$;
while *q not accepted* **do**
    q = new randomQuaternion();
    sum = 0;
    foreach *s in samples* **do**
        sum += $\exp(-\frac{1}{2\sigma}((q_{\mathrm{x}} - s_{\mathrm{x}})^2 + (q_{\mathrm{y}} - s_{\mathrm{y}})^2 + (q_{\mathrm{z}} - s_{\mathrm{z}})^2 + (q_{\mathrm{w}} - s_{\mathrm{w}})^2)$;
    end
    probability = factor * sum;
    if *randomValue.Range(0.0,1.0) <= propability* **then**
        return *q;*
    end
end

CHAPTER 4

Demonstration

4.1 Conditions

CellVIEW [MAPV15] is used to render the scene and Unity3D [1] and C# were used for the development of our tool. With the help of our tool, various macromolecular scenes, consisting of different proteins, can be realized. In Figure 4.1 different kinds of proteins are shown. The shape of the proteins is given by the pdb files and is not changed in the whole modeling process. The domains of the proteins can be distinguished by the color. The user is able to change the color while modeling.



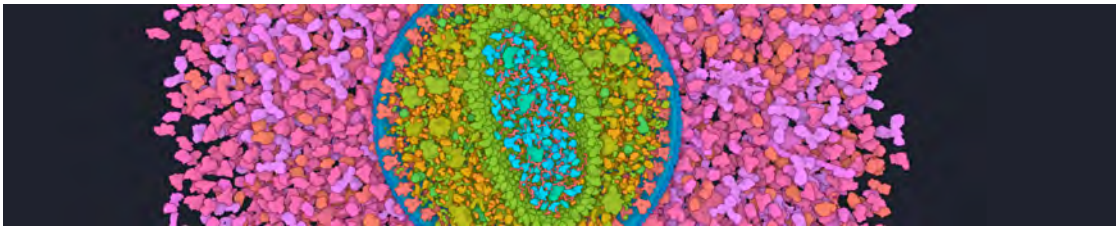
(a) Hemoglobin (b) Immunoglobolin (c) Insulin

Figure 4.1: Different kinds of proteins from the PDB rendered with cellView.

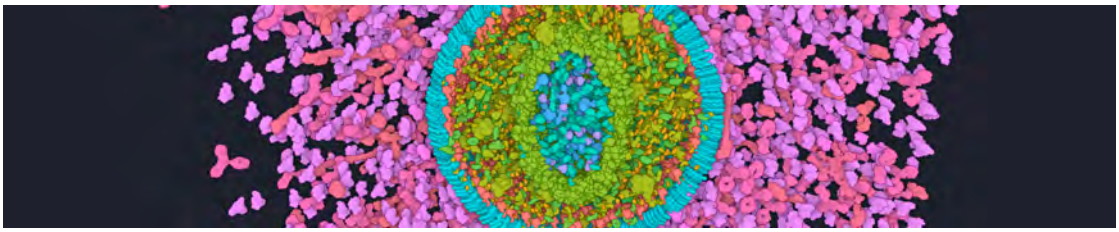
[1] http://www.unity3d.com

4.2 HIV Virion

CellPACK [GTJO14] provides a detailed model of the HIV Virus (shown in Figure 4.2a) rendered with cellVIEW [MAPV15], which we tried to recreate with our statistical modeling approach (shown in Figure 4.2b).



(a) The HIV virus created with cellPack



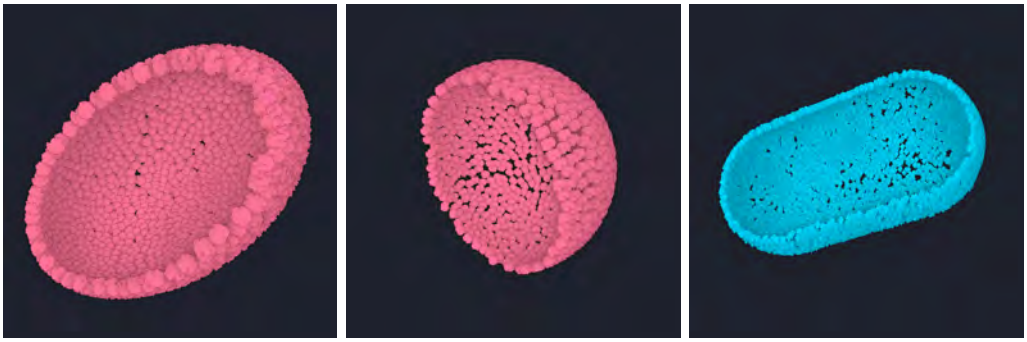
(b) HIV virus statistical approach

Figure 4.2

CellPACK uses a recipe file, to create the HIV virus, which is time consuming and unintuitive. Figure 4.6 shows a section of such a json file. It contains the hierarchy of the virion, which consists of different nested compartments. In the given example, the root and the plasma definition are visible. The compartments contain ingredient descriptions. For example the ingredient with the ID 10 is Albumin, a protein produced in the liver and the most abundant protein in human blood plasma. The ingredient specification contains information about the amount (*nbMol*), the abbreviation for the protein database (*pdb*) and placement information (*center*). The recipe file is not directly used in our approach, but we used it to gather the information about the amount and the location of the ingredients of the HIV to recreate it with the correct composition.

The optical output of the cellPACK model differs from the output of our statistical modeling approach in the following points: The membrane of the HIV virus from cellPACK consists of lipids. We implemented a general approach which can populate membranes with any model. Since we do not obtain a model of a lipid we used as placeholder a protein which shape looks similar to the shape of the lipid used in cellPACK.

We have a general approach for arbitrary distance functions to describe the compartments. As proof of concept the functions for ellipsoids and capsules are implemented, seen in Figure 4.3, since that shapes approximate natural shapes best. The nucleus of the virion from the statistical approach is an ellipsoid. The nucleus from the cellPACK model on

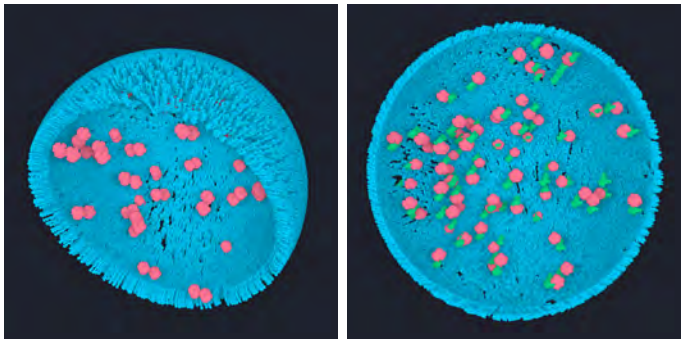


(a) ellipsoid compartment (b) ellipsoid special case (c) capsule compartment

Figure 4.3: Compartments, output of the implemented distance functions

the other hand is a free form surface. The implementation is easily to extend it is just necessary to provide a distance function which describes the free form surface.

Another difference is the overlapping of proteins. In our approach a basic collision detection and a framework to optimize the collision detection are implemented. The result with round proteins is quite good, but for other shapes the surface of the compartment is not populated uniformly and holes appear. The statistical created HIV is modeled without collision detection, therefore overlaps occur which are visible in the output. The output of cellView displays no overlapping. The big advantage of the statistical approach is that the modeling process is considerably faster and more intuitive. Creating the HIV (Figure 4.2a) with cellPACK takes a couple of hours. Modeling the HIV (Figure 4.2b) consisting of the same ingredients, with the statistical approach, takes about 30 minutes.



(a) same proteins (b) different proteins

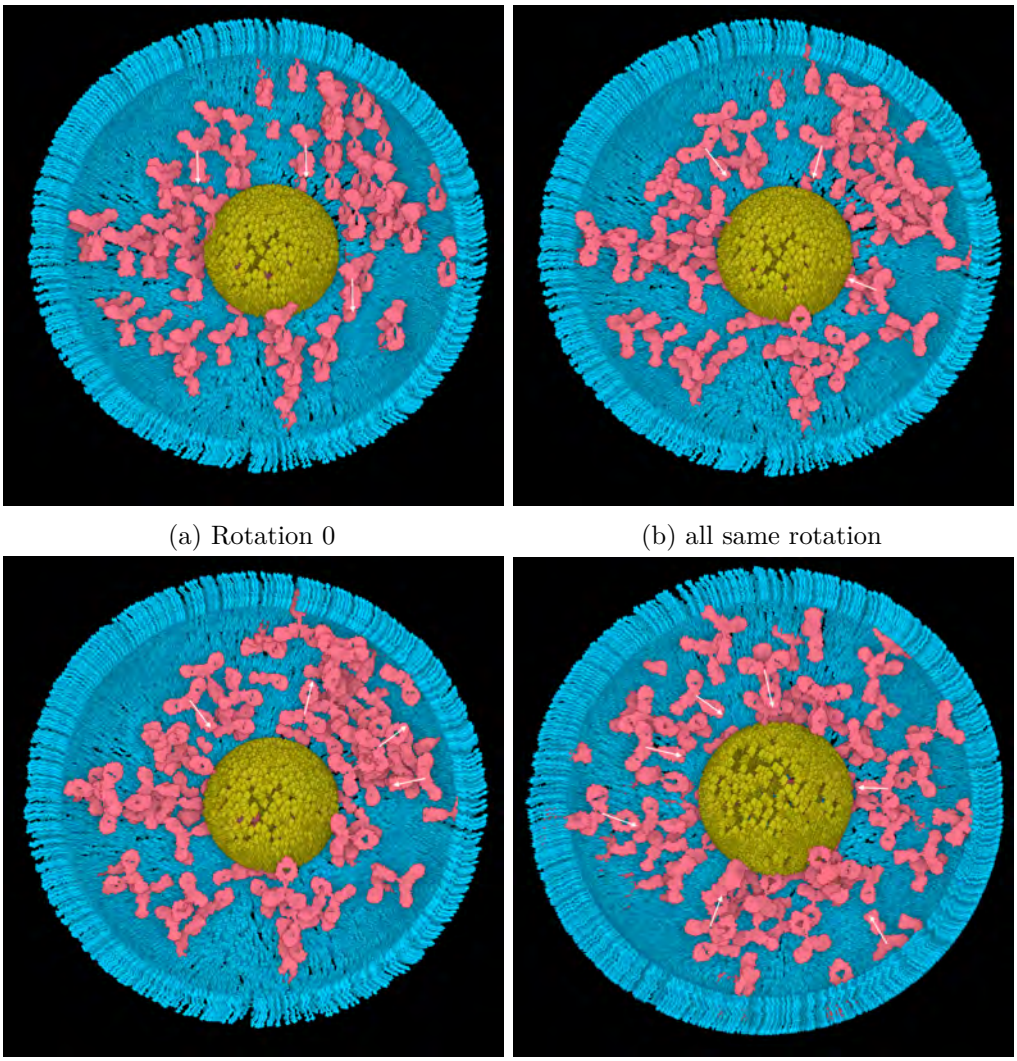
Figure 4.4: Dimer cluster

4.3 Cluster

A cluster consists of an arbitrary amount of proteins. Figure 4.4a shows a compartment with clusters formed by two similar proteins. It can be seen, that some proteins are not clustered since the number of proteins is odd and therefore not every protein has a partner to form a cluster. Figure 4.4b shows clusters that are formed between different proteins. The user can build clusters by moving proteins towards another within a compartment. The rotations of the proteins within a cluster are randomly set. The clusters are distributed randomly in the corresponding compartment object.

4.4 Rotation

When the proteins are added to the scene the rotation quaternion is $[0, 0, 0, 0]$ shown in Figure 4.5a. After the first user affected rotation the rotation is always relative to the surface of the nearest compartment object. In Figure 4.5b all proteins are rotated the same. Figure 4.5c shows the rotation in two different directions with variance 0. The output of the rotation with a variance deviating from zero is shown in Figure 4.5d.



(a) Rotation 0

(b) all same rotation

(c) two different rotations

(d) rotation variance 5

Figure 4.5: (a) shows the output after the proteins were added to the scene and moved into the outer compartment; (b) the output after the first rotation, with a variance of 0; (c) shows the output after rotating one protein in the opposite direction, with a variance of 0; (d) shows the output of the same rotation as (b) but with a variance of 5

Figure 4.6: Part of the HIV recipe file, which describes the structure and ingredients

CHAPTER 5

Discussion and Future Work

Our tool implements the described approach of statistical modeling. The basic concepts are implemented, but there is a great potential in refining and extending the methods, concepts and implementation.

5.1 Cluster

For the cluster formation the first steps are implemented. As can be seen in Figure 4.4a and Figure 4.4b the rotations of the proteins included in a cluster is random and the rotation of the whole cluster too. At the moment the only characteristic is the distance between the protein centers. It is not possible to model the protein cluster of the capsid surface with our approach. Therefore the cluster formation have to be extended so that it can capture not only the closeness of the proteins, but also their geometric structure the position of the whole cluster and the rotation relative to the surface.

Templates are another approach to realize clusters, which is implemented in our tool. It is a proof of concept especially for clusters on membranes. A template consists of an arbitrary amount of similar or different proteins and is a fixed small structures with a center. The tool provides an extra view for template creation and the template information can be stored in a json file. These template files can be imported during the modeling process of a scene. At the moment templates can only be used in the compartment creation phase. It is possible to choose "use template" for the surface population instead of a protein type. The output of a surface populated by a template can be seen in Figure 5.1. Since the collision detection for templates is not implemented yet the surface is not completely populated with templates to give a better demonstration. The advantage of the concept of templates is the possibility to create a small structure and use this like a single protein in the scene. The template creation as well as the following usage in the modeling process has to be improved. At the moment just the

basics and the framework to create and use templates is implemented. It is not possible to move or rotate the template after surface population. Therefore to improve the templates the movement, rotation and integration of templates in the decision tree should be implemented.

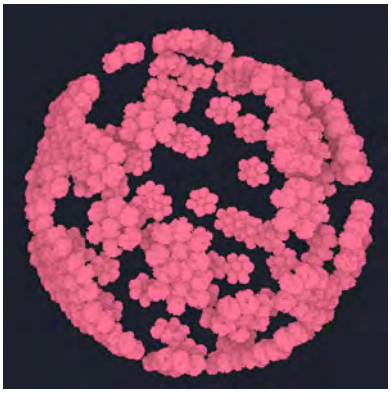


Figure 5.1: Surface populated by a template, to give a better demonstration it is not completely covered

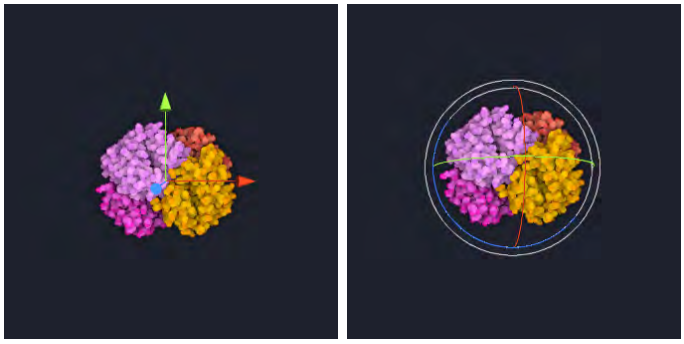
5.2 Rotation

The rotation is implemented as described in Section 3.3. The random quaternions evaluated by the KDE does not constantly deliver visual output of the same quality. One possibility to achieve a better visual output, is to implement different kernels for kernel density estimation and compare which one deliver the best result.

5.3 Compartments

Compartments are described by distance functions which are implemented for ellipsoids and capsules. Free form surfaces would improve the outcome, since capsules and ellipsoids often appear too structured for elements like the nucleus. Basically the program works with any arbitrary distance function, therefor to expand the tool with other shapes only the desired distance function has to be implemented. To use free form surfaces, a distance transform could be used to transform it into a distance field.

At the moment the distribution of the proteins within the compartments or on the membrane is implemented by random samples. To optimize the distribution of the proteins within the compartments different distribution function could be implemented. May be changeable by the user so that the user can choose the distribution with the less artifacts for the current processed form.



(a) move handle (b) rotate handle

Figure 5.2: Hemogolobin, move and rotate handle

Another possibility to increase the usability in relation to compartments is to enable the movement and rotation of compartments after compartment creation with all the enclosed proteins and sub-compartments.

5.4 Usability

We tried to keep the modeling process as simple and intuitive as possible. To move or rotate the proteins there exist two separate handles. The move handle (see Figure 5.2a) is activated by selecting a protein. A click on an already selected protein changes to the rotate-handle (see Figure 5.2b). The accurate placement of objects in 3D space is often slightly difficult. To raise the usability the tool restricts the user to axis aligned movements. Nevertheless the user has to change the perspective repeatedly to place the protein at the desired place. The implementation of another concept or method for the placement in 3D space could improve the usability.

The fact that just protein models are available restricts the user by modeling a biological correct model. It would be a great impact if also lipids and RNA could be used for modeling, which would be possible if appropriate models would be accessible.

CHAPTER 6

Conclusion

The thesis has given an insight into the concepts of our statistical approach for the modeling of microorganisms. The basics as the decision tree, the placement, rotation and clustering of proteins and statistical calculations are implemented in our tool. During the implementation of our approach we found out that the positioning of the proteins in 3D space is one of the most challenging parts of the modeling process, so in the future research this should be addressed. There are many other possibilities to refine and extend the methods, features and implementation, but the result of the approach satisfies the expectation to model a microorganism with several thousand proteins quite simple and fast within one tool that is usable with relative little effort. Our tool shows that it is possible to model organisms as large as the HIV created with cellPACK with reasonable precision. The statistical modeling approach for extensive scenes is very promising, but needs more research, since it is still on the very beginning. Our software is actually a modular framework which can serve as basis for further research.

Bibliography

[AW97] Chidanand Apté and Sholom Weiss. Data mining with decision trees and decision rules. *Future generation computer systems*, 13(2-3):197–210, 1997.

[BDW02] Harry Buhrman and Ronald De Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[BRP05] Ragnar Bade, Felix Ritter, and Bernhard Preim. Usability comparison of mouse-based interaction techniques for predictable 3d rotation. In *International Symposium on Smart Graphics*, pages 138–150. Springer, 2005.

[CCW98] Bark Cheung Chiu and Geoffrey I Webb. Using decision trees for agent modeling: improving prediction performance. *User Modeling and User-Adapted Interaction*, 8(1):131–152, 1998.

[Con03] Darrell Conklin. Music generation from statistical models. In *Proceedings of the AISB 2003 symposium on artificial intelligence and creativity in the arts and science*, pages 30–35, 2003.

[Die06] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

[Grą14] Krzysztof Grąbczewski. *Meta-learning in decision tree induction*. Springer, 2014.

[GTJO14] Ludovic Autin Stefano Forli Michel F. Sanner Graham T. Johnson, David S. Goodsell and Arthur J. Olson. 3d molecular models of whole hiv-1 virions generated with cellpack. *Faraday Discuss.*, 169:23, 2014.

[Han05] Andrew J Hanson. Visualizing quaternions. In *ACM SIGGRAPH 2005 Courses*, page 1. ACM, 2005.

[Lan00] Irene Langkilde. Forest-based statistical sentence generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 170–177. Association for Computational Linguistics, 2000.

[LMMS+16] M Le Muzic, Peter Mindek, Johannes Sorger, Ludovic Autin, David S Goodsell, and Ivan Viola. Visibility equalizer cutaway visualization of mesoscopic biological models. In *Computer Graphics Forum*, volume 35, pages 161–170. Wiley Online Library, 2016.

[MAPV15] Mathieu Le Muzic, Ludovic Autin, Julius Parulek, and Ivan Viola. cellview: a tool for illustrative and multi-scale rendering of large biomolecular datasets. In Katja B"uhler, Lars Linsen, and Nigel W. John, editors, *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 61–70. EG Digital Library, The Eurographics Association, sep 2015.

[Par62] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[Sil86] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[SL91] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.